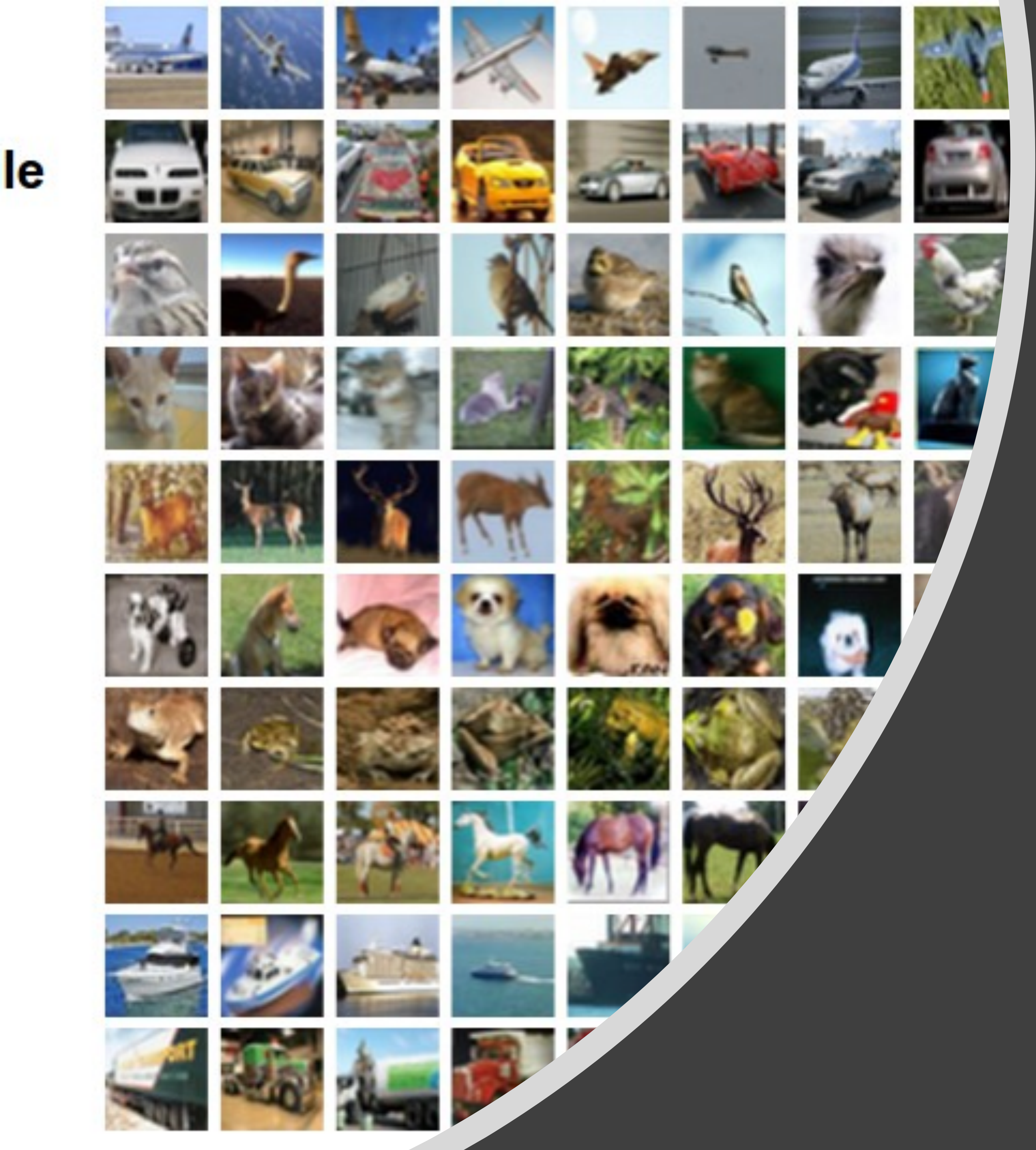


Dynamic Learning

Logistic Regression, Neural Networks and Gradient Descent

Final Project Proposal

- Project overview (3 paragraphs)
- Objectives
  - What is the purpose of the project
  - How will the project be evaluated
- Data to be used
- Learning model
- Libraries

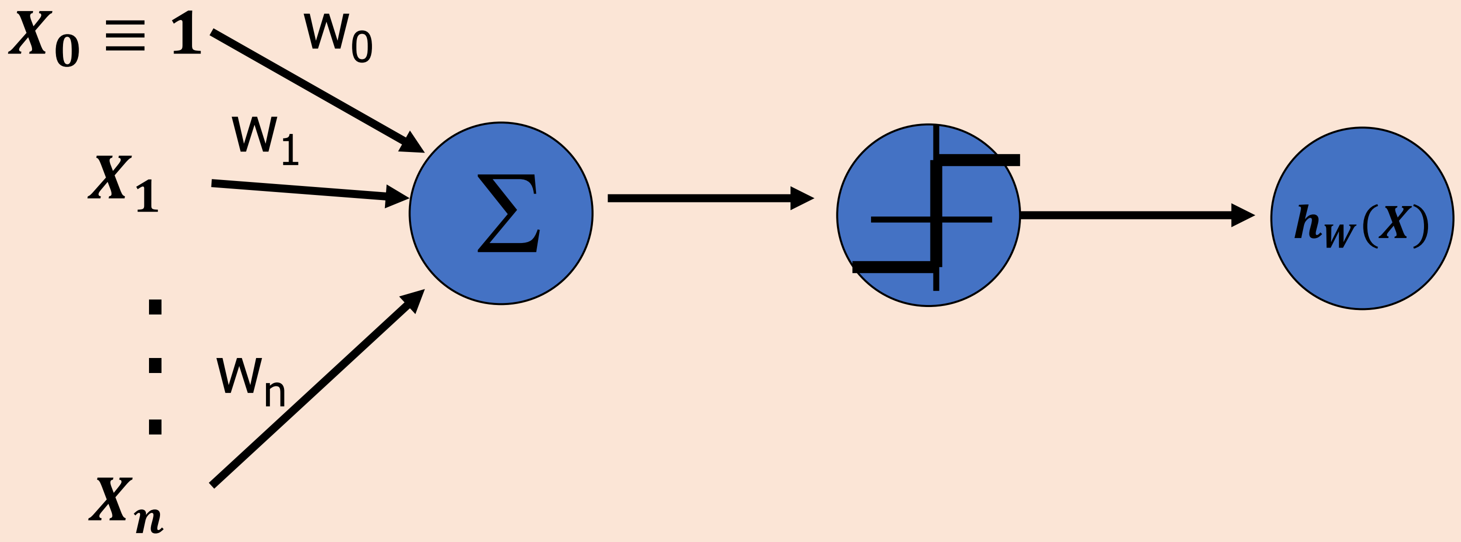
Perceptron Algorithm

- Perceptron Algorithm
  - Multi layered perceptron
- Regression
  - Linear
  - Logistic
- Gradient Descent
  - Batch Gradient Descent
  - Stochastic Gradient Descent
- Introduction to Neural Networks
  - Feedforward Neural Networks

The Perceptron

$$g(z) = sgn(z) = \begin{cases} -1 & z \leq 0 \\ +1 & z > 0 \end{cases}$$

$$h_w(x) = g(\sum w_i x_i) = g(W \cdot X)$$



Perceptron Algorithm

Given a training set $(X_0, y_0), \dots (X_k, y_k)$

$$W \leftarrow (w_{0,1}, \dots w_{0,n})$$

$$for\ (X, y) \quad in \quad (X_0, y_0), \dots (X_k, y_k):$$

$$h \leftarrow g(\sum w_i x_i)$$

$$w_i \leftarrow w_i + \eta \cdot x_i \cdot (y - h)$$

Perceptron learning rule

- The algorithm converges to the correct classification
  - if the training data is linearly separable
  - and η is sufficiently small

- When assigning a value to η we must keep in mind two conflicting requirements
  - Averaging of past inputs to provide stable weights estimates, which requires small η
  - Fast adaptation with respect to real changes in the underlying distribution of the process responsible for the generation of the input vector $\boldsymbol{x}$, which requires large η

Linear Regression

Let $W = (w_0, \ldots w_n), X = (1, x_1 \ldots w_n),$

$$l_W(X) = \sum w_i x_i$$

Given a training set $(X_0, y_0), \ldots (X_k, y_k)$

$$\text{Wopt} = \underset{W}{\text{argmin}} \sum (y_j - l_W(X_j))^2 = argmin_W \left| |y - M \cdot W| \right|^2$$

$$\mathrm{W_{opt}} = (\mathrm{M^T} \cdot M)^{-1} M^T y$$

where $M = [X_0, \ldots, X_k]$

Linear Regression via Gradient Descent

Given a training set $(X, y) \in (X_0, y_0), \ldots (X_k, y_k)$

$$J(W) = \frac{1}{2}\left(y - l(X)\right)^2$$

$$\frac{\partial J(W)}{\partial w_i} = \left(y - l(X)\right) \cdot x_i$$

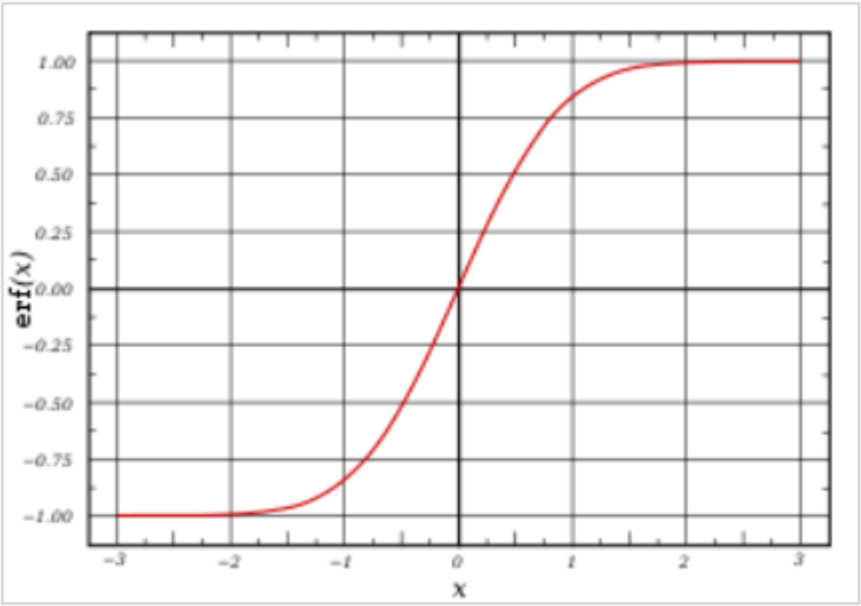
Widrow-Hoff learning rule:

$$\mathrm{w_i} \leftarrow w_i - \eta \cdot \frac{\partial J(W)}{\partial w_i} = w_i - \eta \cdot \left(y - l(X)\right) \cdot x_i$$

Logistic Regression

Sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}}$$



$$h_W(X) = \mathrm{g}(\mathrm{W} \cdot X) = \frac{1}{1 + e^{-W \cdot X}}$$

Gradient Descent

Given a training set $(X_0, y_0), \ldots (X_k, y_k)$

Assume:

$$P\left(y_j = 1 \middle| X_j, W\right) = h_W\left(X_j\right)^{y_j}(1 - h_W\left(X_j\right)^{1-y_j})$$

$$L(W) = \Pi_j P\left(y_j = 1 \middle| X_j, W\right) = \Pi_j h_W\left(X_j\right)^{y_j}(1 - h_W\left(X_j\right)^{1-y_j})$$

$$J(W) = \log L(W) = \sum_j y_j \log(h_W\left(X_j\right) + \left(1 - y_j\right)\log(1 - h_W(X_j)$$

$$\frac{\partial J}{\partial w_i}(X) = (y - h_W\left(X_j\right)x_i$$

using $\frac{\partial g}{\partial z} = g(z)(1 - g(z))$

Gradient Descent

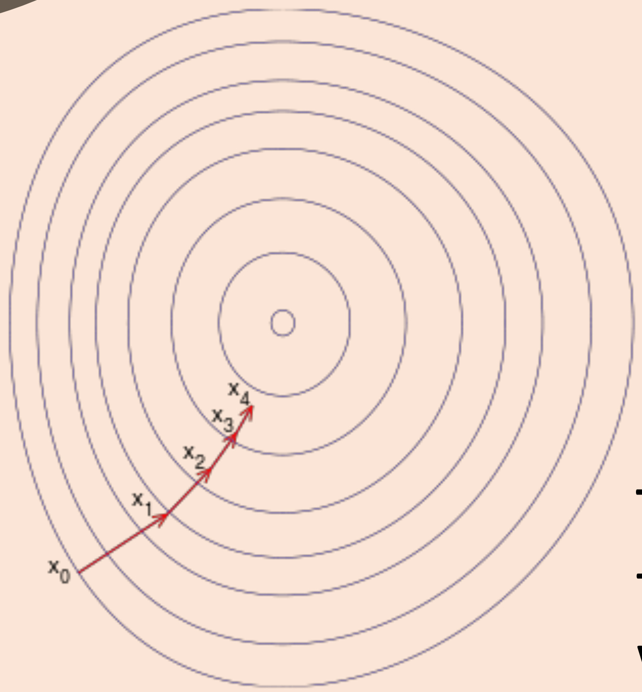
Given a training set $(X, y) \in (X_0, y_0), \dots (X_k, y_k)$

$$W \leftarrow (w_{0,1}, \dots w_{0,n})$$

$$for\ (X, y)\quad in\quad (X_0, y_0), \dots (X_k, y_k):$$

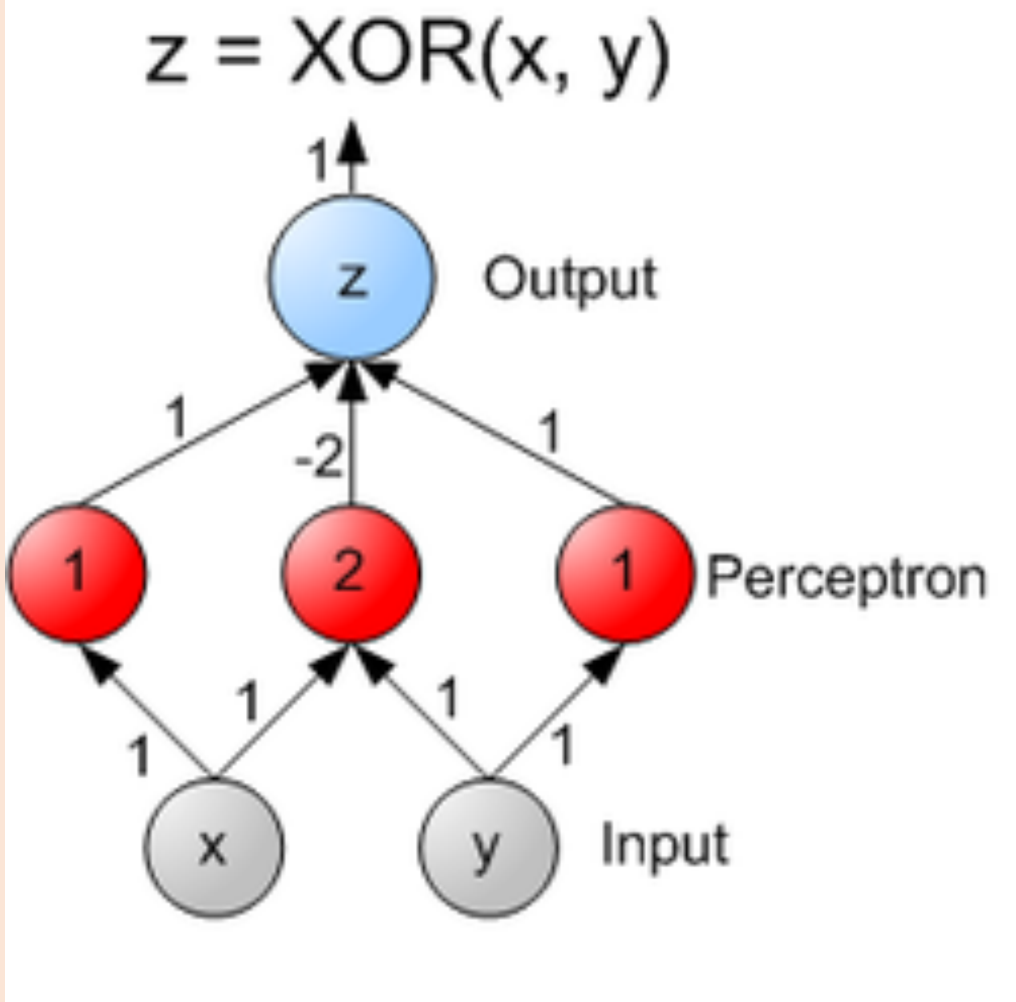
$$h \leftarrow g(\sum w_i x_i)$$

$$w_i \leftarrow w_i + \eta \cdot x_i \cdot (y - h)$$



This shows the update algorithm used for the perceptron would converge to the weights maximizing likelihood

Feedforward Neural Networks



Deep Feedforward Neural Networks